WIND RIVER

Wind River®
General Purpose Platform, VxWorks® Edition

GETTING STARTED

3.3

Copyright © 2006 Wind River Systems, Inc.

All rights reserved. No part of this publication may be reproduced or transmitted in any form or by any means without the prior written permission of Wind River Systems, Inc.

Wind River, the Wind River logo, Tornado, and VxWorks are registered trademarks of Wind River Systems, Inc. Any third-party trademarks referenced are the property of their respective owners. For further information regarding Wind River trademarks, please see:

http://www.windriver.com/company/terms/trademark.html

This product may include software licensed to Wind River by third parties. Relevant notices (if any) are provided in your product installation at the following location: *installDir*/*product_name*/**3rd_party_licensor_notice.pdf**.

Wind River may refer to third-party documentation by listing publications or providing links to third-party Web sites for informational purposes. Wind River accepts no responsibility for the information provided in such third-party documentation.

Corporate Headquarters
Wind River Systems, Inc.
500 Wind River Way
Alameda, CA 94501-1153
U.S.A.

toll free (U.S.): (800) 545-WIND
telephone: (510) 748-4100
facsimile: (510) 749-2010

For additional contact information, please visit the Wind River URL:

http://www.windriver.com

For information on how to contact Customer Support, please visit the following URL:

http://www.windriver.com/support

7 Jun 06
Part #: DOC-15753-ND-00

Contents

1 Overview 1

1.1 Introduction 1

1.2 Installation Media 2

1.3 General Purpose Platform Features 4

1.4 Documentation Guide 5

1.4.1 General Purpose Platform Documentation 5

1.4.2 Component Documentation 6

Operating System Documentation 6
Host Tools Documentation 7
Connectivity Documentation 8

1.4.3 Context Sensitive Help 9

1.4.4 Eclipse Documentation 9

1.4.5 Online Documentation 9

1.4.6 Getting Documents in Hardcopy 10

2 Creating Projects 11

2.1 Introduction 11

Starting Wind River Workbench 12

Starting from the Command Line: Setting Your Environment 12

2.2 VxWorks Boot Loaders 13

2.3 VxWorks Kernel Images 14

2.4 VxWorks Kernel Applications 15

2.5 Real-time Process Applications 16

2.6 Shared Libraries 17

2.7 User-Defined Projects 18

3 Compiling Source 19

3.1 Introduction 19

3.2 Compiling VxWorks OS Source 19

Source Code Limitations 20
Files Supplied as Object Code 20

3.2.1 Building VxWorks Source Code Modules (Workbench) 22

Additional Build Options 23
Deleting the Source Build Project 24

3.2.2 Building VxWorks Source Code Modules (Command Line) 25

Additional Build Options 26

3.2.3 Restoring Original Archives and Object Directories 28

3.2.4 Supported CPU and TOOL Values 28

4 Configuring VxWorks to Include Components 31

4.1 Introduction 31

4.2 Managing Components Using Workbench 32

4.2.1 Browsing the Kernel Configuration Editor 34

4.2.2 Using Find to Locate Components 34

4.2.3 Including or Excluding Components 35

4.3 Managing Components Using a Command-Line Build 36

vxprj Build Method 37
BSP config.h Build Method 37

4.4 Configuring OS Components 38

4.5 Configuring Connectivity Components 38

4.5.1 Wind River Network Stack 38

4.5.2 Wind River PPP 38

4.5.3 Wind River TIPC 42

4.5.4 Wind River USB 42

USB Peripheral Stack Components 42
USB Host Stack Components 43
USB Components: Command-Line Configuration 45

5 Migration 47

5.1 Introduction 47

5.2 Operating System Migration 48

5.3 Host Tool Migration 48

5.3.1 Wind River Workbench 48

5.3.2 Wind River Compiler 48

5.3.3 Wind River GNU Compiler 48

5.3.4 Wind River ScopeTools 49

5.4 Connectivity Component Migration 49

5.4.1 Wind River Network Stack 49

5.4.2 Wind River PPP 49

5.4.3 Wind River TIPC 49

5.4.4 Wind River USB 50

A WDB over TIPC 51

A.1 Introduction 51

A.2 Setup Requirements 52

A.3 Target Configuration 53

A.4 Establishing a Target Server Connection 54

B Removing Run-time Libraries 55

B.1 Introduction 55

B.2 Removing Run-time Libraries from VxWorks Images 55

Index 57

1 Overview

1.1 Introduction 1
1.2 Installation Media 2
1.3 General Purpose Platform Features 4
1.4 Documentation Guide 5

1.1 Introduction

This document provides information about building, configuring, migrating, and locating additional documentation for the Wind River General Purpose Platform, VxWorks Edition.The General Purpose Platform provides an integrated solution including VxWorks 6.3, Wind River Workbench 2.5, a dual IPv4/IPv6 network stack, and additional run-time components.

Terminology and Conventions

The following terms are used in this document:

host
A computer on which the Wind River development tools run.

target
A processor board that runs VxWorks (Wind River's real-time operating system) and applications developed with Wind River Workbench.

target server
A service that runs on the host and manages communications between host tools (such as the VxWorks development shell, debugger, and browser) and the target system itself. One target server is required for each target.

Wind River registry
A Wind River service that keeps track of, and provides access to, target servers. One registry may serve a network, or several registries may run on each host.

The following conventions are used in this document:

1. The root installation directory is identified as *installDir* in this document. The environment variable **WIND_HOME** must be set to the root installation directory for Wind River Workbench to work.
2. A series of items to be selected from the GUI is denoted by **A > B > C**. The elements **A**, **B**, and **C** may be menu items, buttons, or tabs.
3. Pathnames that apply to both UNIX and Windows are shown with forward slashes (/).

1.2 Installation Media

The General Purpose Platform installation is provided on a collection of CD-ROMs. Depending on your specific requirements, you may only need to install a subset of the installation media. This section provides a description of each CD and identifies the CDs that are required for this release.

Required CDs

The following CDs are required for the General Purpose Platform installation:

Wind River Workbench 2.5
This CD installs the host tools and development environment for the General Purpose Platform.

VxWorks 6.3 and General Purpose Technologies
This CD installs the VxWorks operating system and Wind River VxWorks Simulator board support packages (BSPs).

You must install at least *one* of the following compiler CDs (if desired, both CDs can be installed):

Wind River Compiler 5.4.0
This CD installs the Wind River Compiler.

Wind River GNU Compiler 3.4.4 for VxWorks 6.3
This CD installs the Wind River GNU Compiler.

Optional CDs

In addition to the required CDs listed previously, a number of optional CDs are available. These CDs include a number of reference BSPs for the supported architecture families.

NOTE: The optional CDs include reference BSPs only. General architecture support for the development environment, the OS, and compilers is provided on the required CDs.

If you would like to install reference BSPs for one or more architectures, install one or more of the following CDs as appropriate:

Board Support Packages for VxWorks 6.3 ARM and XScale
Installs the VxWorks 6.3 reference BSPs for the ARM and Intel XScale architectures.

Board Support Packages for VxWorks 6.3 Intel Architecture and SuperH
Installs the VxWorks 6.3 reference BSPs for the Intel Architecture and Renesas SuperH architectures.

Board Support Packages for VxWorks 6.3 MIPS
Installs the VxWorks 6.3 reference BSPs for the MIPS architecture.

Board Support Packages for VxWorks 6.3 PowerPC
Installs the VxWorks 6.3 reference BSPs for the PowerPC architecture.

NOTE: The BSP CDs listed above may only contain a subset of the BSPs available for your architecture with this release. Additional BSPs may be available on the Wind River online support site. For more information, see your release notes.

1.3 General Purpose Platform Features

The General Purpose Platform includes a set of development tools and software run-time components, at the core of which is the VxWorks real-time operating system. The General Purpose Platform extends VxWorks by adding networking and communications components, as well as an enhanced set of host tools, including compilers, static analysis, run-time analysis, and debugging tools.

The General Purpose Platform includes the following components:

- VxWorks 6.3, a high-performance, memory protected real-time operating system.
- Wind River VxWorks Simulator 6.3, the VxWorks host-based target simulator.
- Wind River Workbench 2.5, a development suite that facilitates managing and building projects, establishing and managing host-target communication, and running, debugging, and monitoring VxWorks applications. Wind River Workbench also includes the following:
  - Wind River System Viewer 4.8, a software logic analyzer that provides graphical representations of the dynamic interactions of elements of the system.
  - Wind River ScopeTools 5.5, run-time analysis tools including:
    - Wind River MemScope 3.7, an online memory analyzer
    - Wind River ProfileScope 4.9, a statistical profiler
    - Wind River StethoScope 7.9, a real-time data monitor

    And optionally including:

    - Wind River CoverageScope 2.4, a code coverage analysis tool
    - Wind River TraceScope 3.5, an execution-flow trace tool
- Wind River Compiler 5.4, a compiler for C and C++ development.
- Wind River GNU Compiler 3.4.4, a compiler for C and C++ development.
- Wind River Network Stack 3.1, a dual IPv4/IPv6 TCP/IP network stack.
- Wind River PPP 2.2.2, source code for managing protocol components through the Point-to-Point Protocol (PPP). Bundled with Wind River PPP 2.2.2 are PPP over Ethernet (PPPoE), Remote Access Framework (RAF), Multilink PPP (ML-PPP), and Bandwidth Allocation Control Protocol/Bandwidth Allocation Protocol (BACP/BAP) support. These components allow for the establishment, authentication, and control of PPP connections.

- Wind River TIPC 1.5, an implementation of the Transparent Inter Process Communication (TIPC) protocol. This product is a port of the open-source implementation on Linux.
- Wind River USB 2.3, including both a Universal Serial Bus (USB) host stack and a USB peripheral stack. The USB host stack enables you to write your own USB host controller driver. The USB peripheral stack enables you to write your own USB target application or target controller driver.

For complete version information for the components included in the General Purpose Platform, check your installation log file located at *installDir*/**setup.log**.

1.4 Documentation Guide

Documentation for the General Purpose Platform falls into these general categories:

- *1.4.1 General Purpose Platform Documentation*, p.5
- *1.4.2 Component Documentation*, p.6
- *1.4.3 Context Sensitive Help*, p.9
- *1.4.4 Eclipse Documentation*, p.9
- *1.4.5 Online Documentation*, p.9

1.4.1 General Purpose Platform Documentation

The General Purpose Platform documentation includes:

- *Wind River General Purpose Platform, VxWorks Edition Getting Started* (this guide)
- *Wind River General Purpose Platform, VxWorks Edition Release Notes*

The release notes include:

- the latest list of supported hosts and targets
- information on compatibility with older releases
- an outline of new features
- any caveats concerning the current release

1.4.2 Component Documentation

Your release also includes the general component documentation for each feature. This section outlines the documentation available with this release.

NOTE: In some cases, the component release version (see *1.3 General Purpose Platform Features*, p.4) does not match the documented component version shown in this section. For example, this release includes Wind River VxWorks Simulator 6.3. However, the document provided with this release is the *Wind River VxWorks Simulator User's Guide, 6.1*. For additional documentation updates, see the release notes.

Operating System Documentation

Documentation for the VxWorks operating system includes the following guides:

- *VxWorks Application Programmer's Guide, 6.3*
- *VxWorks Kernel Programmer's Guide, 6.3*
- *VxWorks Command-Line Tools User's Guide, 6.2*
- *VxWorks Architecture Supplement, 6.3*
- *VxWorks BSP Developer's Guide, 6.2*
- *VxWorks Device Driver Developer's Guide, 6.2*
- *VxWorks Migration Guide, 6.3*
- *VxWorks Hardware Considerations Guide, 6.0*

In addition to the above guides, the following reference documentation is available:

- VxWorks API References, 6.3
  - *VxWorks Application API Reference*
  - *VxWorks Drivers API Reference*
  - *VxWorks Kernel API Reference Volume 1: Libraries*
  - *VxWorks Kernel API Reference Volume 2: Routines*

  (In addition to HTML, the above API references are available in UNIX-style man pages.)
- VxWorks BSP Reference
- VxWorks Errno Code List

Host Tools Documentation

The following host tool documentation is available with this release:

- **Wind River Workbench**

  Documentation for Wind River Workbench includes:

  – *Wind River Workbench User's Guide, 2.5*
  – *Wind River Workbench Migration Guide, 2.5*
  – *Wind River System Viewer User's Guide, 4.8*
  – *Wind River Workbench User Interface Reference, 2.5*

- **Wind River ScopeTools**

  Documentation for Wind River ScopeTools includes:

  – *Wind River MemScope User's Guide, 3.7*
  – *Wind River ProfileScope User's Guide, 4.9*
  – *Wind River StethoScope for VxWorks User's Guide, UNIX version, 7.9*
  – *Wind River StethoScope for VxWorks User's Guide, Windows version, 7.9*
  – *Wind River CoverageScope for VxWorks User's Guide, 2.4* (optional)
  – *Wind River TraceScope for VxWorks User's Guide, 3.5* (optional)

NOTE: Each Wind River ScopeTools user guide (as listed above) includes a detailed getting started chapter with information that you will find necessary and useful when invoking the tool.

- **Wind River VxWorks Simulator**

  Documentation for the Wind River VxWorks Simulator includes:

  – *Wind River VxWorks Simulator User's Guide, 6.1*

- **Wind River Compiler**

  Documentation for the Wind River Compiler includes:

  – Wind River Compiler User's Guides, 5.4
  – Dinkum C++ Library Reference Manual

- **Wind River GNU Compiler**

  The GNU compiler documentation provided by Wind River is a convenient collection of the Free Software Foundation (FSF) manuals for the GNU C and C++ compiler and its supporting tools: the C preprocessor, assembler, static

linker, binary utilities, and make utility. The documentation included in this release is as follows:

– *Wind River GNU Binary Utilities, 3.4.4*
– *Wind River GNU C Preprocessor, 3.4.4*
– *Wind River GNU Make, 3.80*
– *Wind River GNU Compiler: Using as, 3.4.4*
– *Wind River GNU Compiler: Using the GNU Compiler Collection, 3.4.4*
– *Wind River GNU Compiler: Using ld, 3.4.4*

NOTE: FSF develops software under Solaris/UNIX, and examples in their manuals reflect this. However, the GNU tools do operate reliably under Windows.

- **On-Chip Debugging (Optional)**

  Documentation for the on-chip debugging option includes:

  – *Wind River Workbench On-Chip Debugging Guide, 2.5*
  – *Wind River Workbench On-Chip Debugging Command Reference, 2.5*
  – *Wind River Workbench On-Chip Debugging Configuration Options Reference, 2.5*
  – *Wind River ICE SX for Wind River Workbench Hardware Reference, 2.5*
  – *Wind River Probe for Wind River Workbench Hardware Reference, 2.5*
  – *Wind River Trace for Wind River Workbench Hardware Reference, 2.5*
  – *Wind River Board Bring-up Guide, 1.0*

Connectivity Documentation

Documentation for connectivity components includes:

- *Wind River Network Stack for VxWorks 6 Programmer's Guide, 3.1*
- *Wind River PPP for VxWorks 6 Programmer's Guide, 2.2*
- *Wind River TIPC for VxWorks 6 Programmer's Guide, 1.5*
- *Wind River USB for VxWorks 6 Programmer's Guide, 2.3*
- *Wind River USB for VxWorks 6 API Reference, 2.3*

1.4.3 Context Sensitive Help

Help buttons in Wind River Workbench and Wind River System Viewer provide information on the component you are currently using.

From a Workbench view, pressing **F1** on Windows, **Ctrl-F1** on Linux, or the **Help** button on Solaris opens an infopop containing a brief description of the view, as well as links to related topics in the documentation. Pressing **F1** within an enabled area in System Viewer opens the appropriate help page.

System Viewer has additional ways to access context-sensitive help. Right-clicking a specific event within a tool takes you to the *System Viewer User's Reference Event Dictionary* information for that event or state stipple.

1.4.4 Eclipse Documentation

The *Eclipse Workbench User Guide* and *Eclipse Platform Plug-in Developer Guide* (produced by the Eclipse Foundation, not by Wind River) provide information about general Eclipse concepts and procedures, as well as guidelines for writing your own Eclipse plug-ins.

1.4.5 Online Documentation

This section describes the online documentation available with this release.

Online Manuals

This release includes a subset of the manuals (as described earlier in this chapter) in HTML and PDF format. You can open the online manuals from the **Help > Help Contents** menu in Workbench, or by navigating to your Workbench installation directory, then navigating to *installDir*/**docs/index_component.html**. A full-text search facility is available within the Workbench help browser.

Online Help

See *1.4.3 Context Sensitive Help*, p.9.

Man Pages

UNIX-style man pages for API reference entries are available for Solaris hosts. Type **man** *functionName* to display the man page.

1.4.6 Getting Documents in Hardcopy

The General Purpose Platform ships with the following manuals in hardcopy:

- *Wind River General Purpose Platform, VxWorks Edition Getting Started* (this manual)
- *Wind River General Purpose Platform, VxWorks Edition Release Notes*

Other books associated with this product suite can be ordered from the Wind River Bookstore. Visit the Bookstore at the following URL (support login required):

http://www.windriver.com/windsurf/bookstore

Once you have logged into the support site, click the link for the online bookstore.

NOTE: Print-ready PDFs are available from the title page of the corresponding online document. Navigate to the proper document under **Help > Help Contents > Wind River Documentation**, then click the PDF icon to the right of the book title to access the PDF.

2

Creating Projects

2.1 Introduction 11

2.2 VxWorks Boot Loaders 13

2.3 VxWorks Kernel Images 14

2.4 VxWorks Kernel Applications 15

2.5 Real-time Process Applications 16

2.6 Shared Libraries 17

2.7 User-Defined Projects 18

2.1 Introduction

This chapter provides an overview of types of builds with which you might be familiar, suggests the Wind River Workbench project most suited to each type of build, and provides pointers to additional information.

For detailed information on all build types, see *Wind River Workbench User Interface Reference: Building Projects*.

Starting Wind River Workbench

To start Workbench on Solaris or Linux, enter the following:

```
% cd installDir
% ./startWorkbench.sh
```

To start Workbench on Windows, go to **Start > Programs > Wind River > Workbench 2.5 > Wind River Workbench 2.5** (if you chose the default program group during installation).

For introductory information and a tutorial, see *Wind River Workbench User's Guide*. This manual is available from the Workbench online help system:

1. With Wind River Workbench running, choose **Help > Help Contents**.
2. In the online help table of contents, choose **Wind River Documentation > Guides > Host Tools > Wind River Workbench User's Guide 2.5, VxWorks Version**.

Additional tutorials for this release are available at:

http://www.windriver.com/support

Starting from the Command Line: Setting Your Environment

To use the tools included with your Platform from the command line, you need to configure some environment variables and other settings. The best way to do this is with the **wrenv** environment utility.

- On Solaris or Linux, open a shell and set your environment to access the Wind River host tools and Wind River Compiler.

  In your installation directory, run the following command:

  ```
  % ./wrenv.sh -p vxworks-6.3
  ```

NOTE: If your shell configuration file (**.profile**, **.cshrc**, **.tcshrc**, and so forth) overwrites the environment each time a new shell is created, the above command may not work. If you find that you cannot invoke the Workbench tools after executing the above command, use the following command:

```
% eval 'installDir/wrenv.sh -p platform -o print_env -f shell'
```

where *shell* is **sh** or **csh**, depending on the current shell program. For example:

```
% eval './wrenv.sh -p vxworks-6.3 -o print_env -f sh'
```

- On Windows, select **Start > Programs > Wind River > Workbench 2.5 > VxWorks Development Shell**. In this case, the environment is set automatically when the shell is opened.

For more information about the **wrenv** utility, see *VxWorks Command-Line Tools User's Guide*.

2.2 VxWorks Boot Loaders

VxWorks Boot Loader projects are used to create a VxWorks boot loader (a *boot ROM*) to boot and load a target with a VxWorks kernel.

Workbench Build

You must create and configure a Boot Loader project in order to build a VxWorks boot loader (also known as a boot ROM image) from within Workbench.

- To create a VxWorks Boot Loader project, see *Wind River Workbench User's Guide: Boot Loader Project*.
- To configure and build your VxWorks boot loader, see *Wind River Workbench User Interface Reference: Build Console View* and *Build Properties Dialog*.

NOTE: In this release, you cannot edit boot ROM files from within Workbench.

Command-Line Build

To build a VxWorks boot loader on the command line, navigate to the correct directory for your BSP (for example, *installDir*/**vxworks-6.3/target/config/** *bspName*) and then type **make** *bootLoaderType*, where *bootLoaderType* is the type of boot loader you wish to build. For more information on boot loaders and the available boot loader types, see *VxWorks Kernel Programmer's Guide: Kernel*.

For instructions on how to call make directly, see *VxWorks Command-Line Tools User's Guide: Building Kernel and Application Projects*.

2.3 VxWorks Kernel Images

VxWorks Image projects (VIPs) are used to configure and build a VxWorks kernel image to boot your target. A VIP can be a complete application and can also contain projects of other types such as file system projects.

Workbench Build

To build a VxWorks kernel image within Workbench, you must create and configure a VxWorks Image project.

- For instructions about creating VxWorks Image projects, see *Wind River Workbench User's Guide: VxWorks Image Projects*.
- For configuration and build guidelines, see *Wind River Workbench User Interface Reference: Build Console View* and *Build Properties Dialog*.

VxWorks Image Project Options

During VIP creation, the project creation wizard offers you the choice to include one or more options through a check box selection list. The three options are:

- **Use IPv6 enabled kernel libraries**

  This option includes the binary IPv6-enabled kernel libraries.

- **Use System Viewer free kernel libraries**

  This option includes libraries that are built without Wind River System Viewer instrumentation.

- **Use source mode build**

  This option is associated with the VxWorks scalability profiles and is *not* required for a standard project build. (For more information on scalable OS profiles, see *VxWorks Kernel Programmer's Guide: Kernel*.)

CAUTION: All of the kernel components included in your project must support one of the scalability profiles. If not, your build will revert from a source build to a binary build; Workbench will *not* issue a warning or error message for this change.

Building a VxWorks ROMFS File System Project

A VxWorks ROMFS File System project allows you to link a compiled-in file system with your VxWorks kernel image.

For information about creating file system projects, see *Wind River Workbench User's Guide: VxWorks ROMFS File System Projects*.

Command-Line Build

If you are familiar with building a VxWorks kernel image using the command-line build, you can continue to use this technique.

- For general build information, see *VxWorks Command-Line Tools User's Guide: Building Kernel and Application Projects*.

NOTE: The **wrconfig** utility is not available in this release.

- For details about build commands, see the **vxprj** and **cmpscriptLib** reference pages.

  These are available on Solaris by typing **man vxprj** and **man cmpscriptLib**, and on all host platforms from **Help > Help Contents > Wind River Documentation > References > Host Tools > Wind River Project Configuration Command Line API Reference**.

Adding a ROMFS Target File System to the Kernel Image

You can also use **vxprj** to build a ROMFS target file system into your VxWorks kernel image. For details, see *VxWorks Command-Line Tools User's Guide: Working with Projects and Components*.

2.4 VxWorks Kernel Applications

VxWorks Downloadable Kernel Module (DKM) projects are used to manage and build modules that will exist in the kernel space. DKM projects provide an infrastructure for loading, unloading, running, and debugging downloadable kernel modules and kernel applications.

Workbench Builds

To build a VxWorks kernel application from within Workbench, you must create and configure a Downloadable Kernel Module project.

- For details about creating Downloadable Kernel Module projects, see *Wind River Workbench User's Guide: VxWorks Downloadable Kernel Module Projects*.
- To configure and build your kernel module, see *Wind River Workbench User Interface Reference: Build Console View* and *Build Properties Dialog*.

Command-Line Builds

If you are familiar with building application **.o** files to be loaded into the kernel using the command-line build (as was done in Tornado 2.*x*), you can continue to use this technique, but you must create a Downloadable Kernel Module project with your application sources and build your project in Workbench at least once (see above). Then, you can build your kernel module on the command line.

For information about building your kernel module on the command line, see *VxWorks Command-Line Tools User's Guide: Building Kernel and Application Projects*.

2.5 Real-time Process Applications

VxWorks Real-time Process (RTP) projects are used to manage and build modules that will exist outside of the kernel space. This project type allows you to build, run, and debug VxWorks RTP executables.

NOTE: RTP application binaries compiled with earlier versions of VxWorks 6.*x* are not compatible with VxWorks 6.3. This is because of changes in the system call number field as well as changes required for POSIX PSE52 compliance. All RTP applications must be recompiled with VxWorks 6.3.

Workbench Builds

To build an RTP application from within Workbench, you must create and configure a Real-time Process project.

When building an RTP that will use a shared library, the shared library must be explicitly added to the RTP.

- For information about creating RTP projects, see *Wind River Workbench User's Guide: VxWorks Real-time Process Projects*.

- To configure and build your RTP, whether standalone or to use a shared library, see *Wind River Workbench User's Guide: RTPs and Shared Libraries from Host to Target*.

Command-Line Builds

To build an RTP on the command line, you must first create an RTP project with your RTP sources and build your project in Workbench at least once (see above). Then, you can build your RTP on the command line.

When building an RTP that will use a shared library, the shared library must be explicitly added to the RTP.

For information about building RTPs on the command line, see *VxWorks Command-Line Tools User's Guide: Building Kernel and Application Projects*.

2.6 Shared Libraries

VxWorks Shared Library projects are used for libraries that are dynamically linked to VxWorks Real-time Process projects at run-time or to create subprojects that are statically linked to other project types at build time.

Workbench Builds

To build a shared library from within Workbench, you must create and configure a VxWorks Shared Library project.

- For information about creating Shared Library projects, see *Wind River Workbench User's Guide: VxWorks Shared Library Projects*.
- For information about configuring and building a shared library, see *Wind River Workbench User's Guide: RTPs and Shared Libraries from Host to Target*.

Command-Line Builds

To build a shared library on the command line, you must first create a VxWorks Shared Library project with your library sources and build your project in Workbench at least once (see above). Then, you can build your shared library on the command line.

For information about building shared libraries on the command line, see *VxWorks Command-Line Tools User's Guide: Building Kernel and Application Projects*.

2.7 User-Defined Projects

Wind River Workbench provides support for User-Defined projects. With this type of project, you must provide and maintain your own build system, file system management, and so forth. However, the Workbench user interface provides support for tasks such as configuring your build utility and creating build targets using the Workbench GUI. (The VxWorks source code build is an example of a User-Defined project.)

For more information on User-Defined projects, see *Wind River Workbench User's Guide: VxWorks User-Defined Projects*.

Run-time Libraries (Source Build)

The VxWorks source code build is an example of a User-Defined project. If you are entitled to the VxWorks source code product, you can build all OS and component run-time source files that are distributed with the product. There are some **.o** files, however, that cannot be built due to licensing restrictions.

For details, see *3.2 Compiling VxWorks OS Source*, p.19.

3 Compiling Source

3.1 Introduction 19
3.2 Compiling VxWorks OS Source 19

3.1 Introduction

This chapter covers the correct use of the VxWorks source code, if you are entitled to the source code. VxWorks source code is provided under a source code license agreement that provides legal terms and conditions of use. See your agreement (or your Wind River representative) for further information.

3.2 Compiling VxWorks OS Source

VxWorks source code is provided with this release. It is not necessary to compile the source before using the General Purpose Platform. However, with the source code, you can better understand internal kernel functionality and extend your troubleshooting capabilities.

NOTE: The instructions in this section apply to both kernel and user builds unless otherwise noted.

Source Code Limitations

There are some limitations and restrictions you must be aware of when recompiling the VxWorks source:

- The source code must be built from the VxWorks 6.3 installation tree with the host tools and makefiles provided.
- There may be certain portions of the VxWorks object code for which source code has not been provided (see *Files Supplied as Object Code*, p.20).
- For the unmodified source code that is included on the CD, the resulting binaries built using the Wind River Compiler should match the binaries distributed on the CD.
- Modifications to the source code (when permitted) may not be covered by Wind River Customer Support.

You can build the VxWorks source using Workbench or from the command line.

Files Supplied as Object Code

The VxWorks source code does not include source files that have been licensed by Wind River from third parties under restriction and therefore cannot be redistributed in source code format. Table 3-1 lists, for each architecture, the files supplied as object code.

NOTE: The build procedure documented in this chapter copies the relevant **.o** files into your source build directory.

Table 3-1 **Files Supplied as Object Code Only**

Platform	Files		
ARMARCH5, ARMARCH6, XSCALE	**_x_ads_basic.o**	**_x_ads_dmul_mull.o**	**_x_ads_fflt_clz.o**
	_x_ads_d2f.o	**_x_ads_drem.o**	**_x_ads_ffltll_clz.o**
	_x_ads_daddsub_clz.o	**_x_ads_drnd.o**	**_x_ads_fgeqf.o**
	_x_ads_dcheck.o	**_x_ads_dsqrt.o**	**_x_ads_fleqf.o**
	_x_ads_dcmpin.o	**_x_ads_dunder.o**	**_x_ads_fmul_mull.o**
	_x_ads_dcmp.o	**_x_ads_except.o**	**_x_ads_fnorm2_clz.o**
	_x_ads_ddiv.o	**_x_ads_f2d.o**	**_x_ads_fpconst.o**
	_x_ads_deqf.o	**_x_ads_faddsub_clz.o**	**_x_ads_fpinit.o**
	_x_ads_dfixll.o	**_x_ads_fcheck.o**	**_x_ads_frem.o**
	_x_ads_dfix.o	**_x_ads_fcmp.o**	**_x_ads_frnd.o**
	_x_ads_dfixull.o	**_x_ads_fdiv.o**	**_x_ads_fsqrt_clz.o**
	_x_ads_dfixu.o	**_x_ads_feqf.o**	**_x_ads_funder.o**
	_x_ads_dflt_clz.o	**_x_ads_ffixll.o**	**_x_ads_istatus.o**
	_x_ads_dfltll_clz.o	**_x_ads_ffix.o**	**_x_ads_retnan.o**
	_x_ads_dgeqf.o	**_x_ads_ffixull.o**	**_x_ads_status.o**
	_x_ads_dleqf.o	**_x_ads_ffixu.o**	**_x_ads_trapv.o**
PowerPC8xx, 85xx, 403, 405, 440	**arc32.o**	**fpcmp.o**	**gccMathLib.o**
	ceil32.o	**fpexp32.o**	**hyp32.o**
	dccMathLib.o	**fpfloo32.o**	**log32.o**
	dp32.o	**fphyp32.o**	**mod32.o**
	dpcmp.o	**fplog32.o**	**pow32.o**
	exp32.o	**fpmod32.o**	**sqrt32.o**
	floor32.o	**fppow32.o**	**trig32.o**
	fp32.o	**fpsqrt32.o**	
	fparc32.o	**fptrig32.o**	
	fpceil32.o	**func32.o**	
PowerPC60*x*	**atan.o**	**sqrt.o**	**trig.o**
	pow.o		
All Pentium BSPs	**aic7880Lib.o**		

3.2.1 Building VxWorks Source Code Modules (Workbench)

This section describes how to build the VxWorks OS source using Workbench.

Step 1: Back Up the VxWorks Archives

In a host shell, go to the library subdirectory of the target directory. Make a copy of the VxWorks 6.3 archive directory and files for the architecture you want to rebuild. For example, to back up your PowerPC files, type:

```
% cd installDir/vxworks-6.3/target/lib
% mkdir ppcBackup
% cp -r ppc ppcBackup/ppc
% cp libPPC*.a ppcBackup
% cp -r objPPC* ppcBackup
```

Step 2: Create a User-Defined Project

1. Open Workbench and select **File > New > User-Defined Project**.
2. From the **Target Operating System and Version** list, select **Wind River VxWorks 6.3**, then click **Next**.
3. Provide a name for this project.
4. Select **Create project at external location**.
5. Click Browse, then navigate to (or enter):

   installDir/**vxworks-6.3/target/src**

NOTE: For user builds, navigate to *installDir*/**vxworks-6.3/target/usr/src**.

6. Click **Next**. The **Build Support** page appears.
7. Ensure that **User-defined build** option is selected, then enter the following as your **Build Command:**

   make CPU=*cpuType* **TOOL=***toolChain*

   For example:

   ```
   make CPU=PPC32 TOOL=diab
   ```

NOTE: To identify your CPU and associated primary compiler, see Table 3-2.

8. Click **Finish**.

Step 3: Rebuild the VxWorks Source

In the Project Navigator, right-click your project and select **Build Project**.

Step 4: Rebuild with a Secondary Compiler (optional)

In most cases, your primary compiler is the Wind River Compiler (**diab**). If you intend to build all of your projects using this compiler, you do not need to rebuild with the secondary compiler (typically, the Wind River GNU Compiler). However, if you plan to build with **INCLUDE_GCC_INTRINSICS** in your project (or if you intend to build your project with the secondary compiler as listed in Table 3-2), you must build your source project with the secondary compiler in order to include the necessary information for future builds.

NOTE: If you are building the user-side source, you do not need to perform a rebuild with the secondary compiler. Only the Wind River Compiler is supported for user-side builds.

You can change the build command by changing the values of the user build arguments.

1. In the **User Build Arguments** field at the top of the Project Navigator, add the secondary compiler information. The information you type here is appended to the build command you created. For example, if you enter **TOOL=gnu** as a user build argument, the build command becomes:

   ```
   make -f Makefile CPU=PENTIUM2 TOOL=diab TOOL=gnu
   ```

2. In the Project Navigator toolbar, click the **Build** button to start building the project for your CPU and associated secondary compiler.

After a successful build with the secondary compiler, clear the **User Build Arguments** field to revert to the primary compiler value.

Additional Build Options

This section describes additional source build options that you may find useful, depending on your application. These build types are optional and the build steps described in this section are not required to build the basic VxWorks OS source as described previously.

Building IPv6 Network Stack Archives

The basic VxWorks OS source build instructions provided earlier in this section build only the IPv4 network stack archives.

To include an alternate IPv4/IPv6 network stack archive in addition to the default IPv4 archive, add **OPT=-inet6** to the **User Build Arguments** field.

In the **Project Navigator** toolbar, click the **Build** button to start building the IPv6 libraries.

NOTE: It is likely that the build process will produce several build warnings.

For more information about the IPv4 and IPv6 network stacks, see the *Wind River Network Stack for VxWorks 6 Programmer's Guide*.

Building Archives without System Viewer Instrumentation

Specifying the **Use System Viewer free kernel libraries** option when creating a VxWorks Image project creates a project that will build a VxWorks image without System Viewer instrumentation.

However, for this to work, you must have kernel source and must have previously compiled the kernel using the **OPT=-fr** option build (or **OPT=-inet6_fr**, if you want IPv6 support). This option places the System Viewer-free archives in an alternate location (for example, **libwind-fr.a** instead of **libwind.a**, **libnet-fr.a** instead of **libnet.a**, and so on).

The free build, in addition to undefining **WV_INSTRUMENTATION** and **INCLUDE_WNNETD**, also defines a macro **_FREE_VERSION** that causes omission of some debug/sanity checks in performance-critical code (primarily in the networking code).

To build the kernel without System Viewer functionality using Workbench, add **OPT=-fr** or **OPT=-inet6-fr** to the **User Build Arguments** field and build your source project.

Deleting the Source Build Project

If you choose to delete your source build project, do not delete the contents of *installDir*/**vxworks-6.3**. When prompted to confirm the deletion, be sure the **Do not delete contents** radio button is selected (this is the default option) before continuing with the project deletion.

3.2.2 Building VxWorks Source Code Modules (Command Line)

This section describes how to build the VxWorks OS source from the command line.

Step 1: Set Your Command Environment

If you have not already done so, you must set the appropriate environment variables before building your source. The easiest way to do this is with the **wrenv** utility, for example:

```
$ wrenv.sh -p vxworks-6.3
```

or

```
C:\> wrenv.exe -p vxworks-6.3
```

The **wrenv** utility properly sets the necessary environment variables and prepares your environment for VxWorks development. For more information, see *VxWorks Command-Line Tools User's Guide*.

On Windows, you can use the Start menu shortcut as follows: **Start > Programs > Wind River > Workbench 2.5 > VxWorks Development Shell**.

NOTE: If you want to disable the GNU dependency in any BSP, modify the **config.h** file for the BSP to include the following line:

```
#undef INCLUDE_GNU_INTRINSICS
```

Step 2: Back Up the VxWorks Archives

In a host shell, go to the library subdirectory of the target directory. Make a copy of the VxWorks archive directory and files for the architecture you want to rebuild. For example, to back up your PowerPC files, type:

```
% cd installDir/vxworks-6.3/target/lib
% mkdir ppcBackup
% cp -r ppc ppcBackup/ppc
% cp libPPC*.a ppcBackup
% cp -r objPPC* ppcBackup
```

Step 3: Build the Source Code

Go to *installDir*/**vxworks-6.3/target/src** and start the build by invoking the **make** command to build the sources for your **CPU** and **TOOL**.

The syntax for the **make** command is:

```
% cd installDir/vxworks-6.3/target/src
% make CPU=cpuName TOOL=primaryCompilerName
```

If you have **INCLUDE_GCC_INTRINSICS** in your project (or intend to build your project with the GNU compiler) you need to run **make** twice for each CPU—once for the primary compiler and once for the secondary compiler. To identify your CPU and associated primary and secondary compilers, see Table 3-2.

NOTE: If you do not need **INCLUDE_GCC_INTRINSICS** (which allows you to load **gnu**-built objects onto a **diab**-built image) or if you are performing a user-side build, do not run the secondary build.

To build for your secondary compcompiler, the syntax is the same as for the primary compiler. For example, run both of the following:

```
% make CPU=PPC32 TOOL=diab
% make CPU=PPC32 TOOL=gnu
```

NOTE: You can build a debug version of the source by providing a **-g** flag against **ADDED_CFLAGS** and **ADDED_C++FLAGS** in *installDir*/**vxworks-6.3/target/src/Makefile**. For example:

```
make CPU=cpuName TOOL=compilerName ADDED_CFLAGS+=-g ADDED_C++FLAGS+=-g
```

Additional Build Options

This section describes additional source build options that you may find useful, depending on your application. These build types are optional and the build steps described in this section are not required to build the basic VxWorks OS source as described previously.

Building IPv6 Network Stack Archives

The basic VxWorks OS source build instructions provided earlier in this section build only the IPv4 network stack archives.

To include an alternate IPv4/IPv6 network stack archive in addition to the default IPv4 archive, run **make** as follows:

```
make CPU=cpuName TOOL=compilerName OPT=-inet6
```

For example:

```
% make CPU=PPC32 TOOL=diab OPT=-inet6
```

NOTE: You must build both the IPv4 and IPv6 network stack archives to have a working IPv6 image.

For more information about the IPv4 and IPv6 network stacks, see the *Wind River Network Stack for VxWorks 6 Programmer's Guide*.

Building Archives without System Viewer Instrumentation

Specifying the **Use System Viewer free kernel libraries** option when creating a VxWorks Image project creates a project that will build a VxWorks image without System Viewer instrumentation.

However, for this to work, you must have kernel source and must have previously compiled the kernel using the **OPT=-fr** option build (or **OPT=-inet6_fr**, if you want IPv6 support). This option places the System Viewer-free archives in an alternate location (for example, **libwind-fr.a** instead of **libwind.a**, **libnet-fr.a** instead of **libnet.a**, and so on).

The free build, in addition to undefining **WV_INSTRUMENTATION** and **INCLUDE_WNNETD**, also defines a macro **_FREE_VERSION** that causes omission of some debug/sanity checks in performance-critical code (primarily in the networking code).

To do a free build:

1. Do a default build by running **make** as follows:

   ```
   cd target/src
   make CPU=cpuName TOOL=compilerName
   ```

2. Then do a free build by running make again with the **OPT=-fr** or **OPT=-inet6_fr** option:

   ```
   make CPU=PENTIUM3 TOOL=diab OPT=-fr
   ```

NOTE: Some of the required kernel sources for a full kernel build may not be provided to you because Wind River may not have rights to distribute this source code. The objects corresponding to these absent source files will not be built, and so will not be present in the **libarch-fr.a** archive, causing the subsequent image build to fail.

To work around this problem, extract the missing math objects from the default archives and archive them into **libarch-fr.a** or **libarch-inet6_fr.a**.

CAUTION: Do not delete **libarch.a** or any other archives containing objects that cannot be built from source available to you.

3.2.3 Restoring Original Archives and Object Directories

In the VxWorks development shell, go to the library subdirectory of the target directory. Move the recompiled version of the VxWorks archive directory by renaming it, and then restore the original archive directory from the copy you made in Step 2.

For example:

```
% cd installDir/vxworks-6.3/target/lib
% mv ppc ppcRef
% mv ppcBackup/ppc ppc
% mv ppcBackup/libPPC*.a .
% cp -rf ppcBackup/objPPC* .
```

In this example, all of the newly recompiled files are now located in the *installDir*/**vxworks-6.3/target/lib/ppcRef** directory.

3.2.4 Supported CPU and TOOL Values

The source tree build system has been designed and tested to compile source code with a primary compiler only. Table 3-2 lists the primary compiler that Wind River uses to build the VxWorks image for each architecture. The secondary compiler is provided for the application level only, but some run-time support is required. Therefore, the source tree build system builds only the directories necessary to support the secondary compiler.

NOTE: User (RTP) builds support the CPU options listed in Table 3-2. However, only the Wind River Compiler (**diab**) is supported.

Table 3-2 lists both the primary and secondary compilers by architecture.

Table 3-2 **CPU and TOOL Values by Architecture**

Architecture	CPU	Primary Compiler	Secondary Compiler
ARM Architecture Version 5 Processors	ARMARCH5	diab	gnu
ARM Architecture Version 5 Processors (big-endian)	ARMARCH5	diabbe	gnube
ARM Architecture Version 6 Processors	ARMARCH6	diab	gnu

Table 3-2 **CPU and TOOL Values by Architecture** (cont'd)

Architecture	CPU	Primary Compiler	Secondary Compiler
Intel XScale	XSCALE	diab	gnu
PowerPC 60x, 7xx, 74xx, 82xx	PPC32	diab	gnu
PowerPC 40x, 440, 8xx, 85xx	PPC32	sfdiab	sfgnu
MIPS32 BE	MIPS32	sfdiab	sfgnu
MIPS32 LE	MIPS32	sfdiable	sfgnule
MIPS64 BE	MIPS64	diab	gnu
MIPS64 LE	MIPS64	diable	gnule
Pentium	PENTIUM	diab	gnu
Pentium	PENTIUM2	diab	gnu
Pentium	PENTIUM3	diab	gnu
Pentium	PENTIUM4	diab	gnu
SH 7750, 7770	SH7750 (SH32 for RTP builds)	diab	gnu
SH 7750, 7770 LE	SH7750 (SH32 for RTP builds)	diable	gnule
SIMNT	SIMNT (SIMPENTIUM for RTP builds)	diab	gnu
SIMSOLARIS	SIMSPARCSOLARIS	diab	gnu
SIMLINUX	SIMLINUX (SIMPENTIUM for RTP builds)	diab	gnu

NOTE: For **SIMNT**, **SIMSPARCSOLARIS**, and **SIMLINUX**, the **libprocfs.a**, **libtffs.a**, **libusb.a**, and **libusb2.a** libraries are not supported; although they are built from source, they are not present on the distributed CD.

4

Configuring VxWorks to Include Components

4.1 Introduction 31

4.2 Managing Components Using Workbench 32

4.3 Managing Components Using a Command-Line Build 36

4.4 Configuring OS Components 38

4.5 Configuring Connectivity Components 38

4.1 Introduction

This chapter provides information for including each run-time component in a VxWorks image or VxWorks Image project (VIP). In addition, this chapter provides basic configuration information for each component. In some cases, the configuration of a specific component is beyond the scope of this book and a cross-reference to additional documentation is provided.

Configuring Your VxWorks Image

You can configure your VxWorks image to include components in a variety of ways: you can use Workbench to include components into a VxWorks Image project, you can use the **vxprj** command-line build facility, or you can manually edit your BSP configuration files to include or exclude components and then build VxWorks from the command line.

NOTE: This chapter does not include specific instructions for creating a VxWorks Image project or for configuring a kernel image using the command line (or by editing your BSP). If you are unfamiliar with this project type and process, an overview—including references to additional documentation—is available in *2.3 VxWorks Kernel Images*, p.14.

When choosing a method for configuring and building your image, consider the following criteria:

- Configuring and building a VxWorks image using Workbench or the **vxprj** command-line build facility provides an easy and accurate method of adding needed components and ensuring required dependencies are also included.
- Configuring and building a VxWorks image using the BSP configuration method involves editing text files that contain component listings and parameters for initialization, and calling the make utility to build a system image. This process affords you the ability to automate the generation and build of your VxWorks image, but it *requires* that you manage component dependencies.

NOTE: This document describes the general method for customizing a BSP to include components. Detailed component-level configuration macros and descriptions are documented in the respective component documentation (see *1.4 Documentation Guide*, p.5).

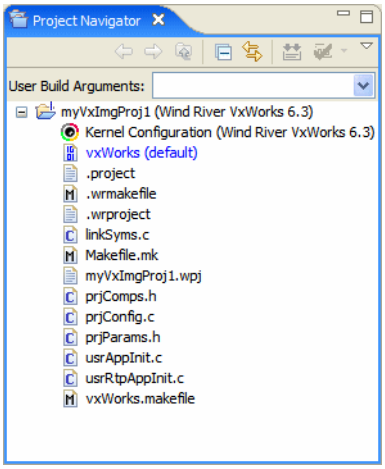
4.2 Managing Components Using Workbench

Each of the component inclusion procedures described in this chapter is relative to the **Components** tab of the Kernel Configuration Editor (see Figure 4-2). To view the Kernel Configuration Editor, you must have a VxWorks Image project open in your workspace (for more information on VxWorks Image projects, see *2.3 VxWorks Kernel Images*, p.14).

To display the Kernel Configuration Editor:

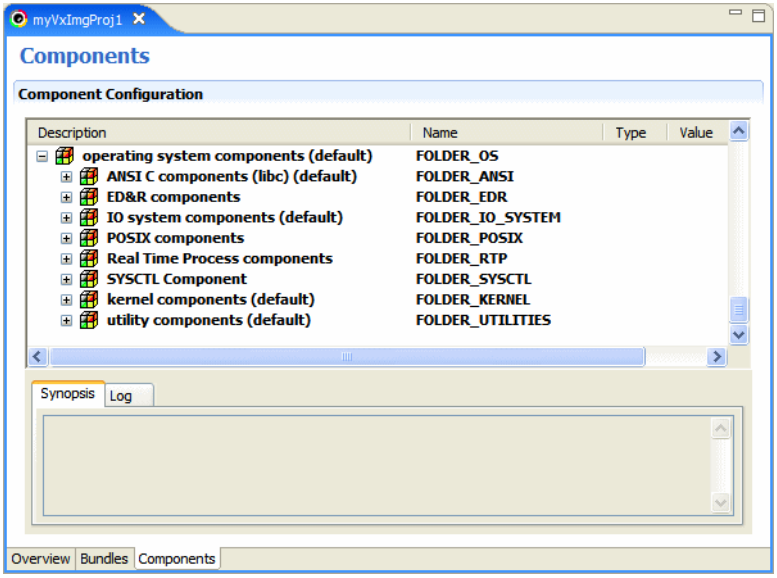
1. In the Workbench Project Navigator, with your VxWorks Image project expanded, double-click **Kernel Configuration** (Figure 4-1).

Figure 4-1 **Viewing the Kernel Editor**



2. When the Kernel Configuration Editor displays, select the **Components** tab (Figure 4-2).

Figure 4-2 **Kernel Configuration Editor Components Tab**



There are a variety of ways in which you can select components to include in your VxWorks Image project. The following sections include some examples.

4.2.1 Browsing the Kernel Configuration Editor

The **Components** tab of the Kernel Configuration Editor displays a listing of components grouped by functionality.

- **Bold Text**—identifies components that are already included in your project.
- Regular Text—identifies components *not* included in your project.
- *Italic Text*—identifies components that are not available for use in your project.

  Components (and subcomponents) may not be available for use for a variety of reasons, including:

  – The component source may not have been compiled.

  – The component may not be supported for use with your platform.

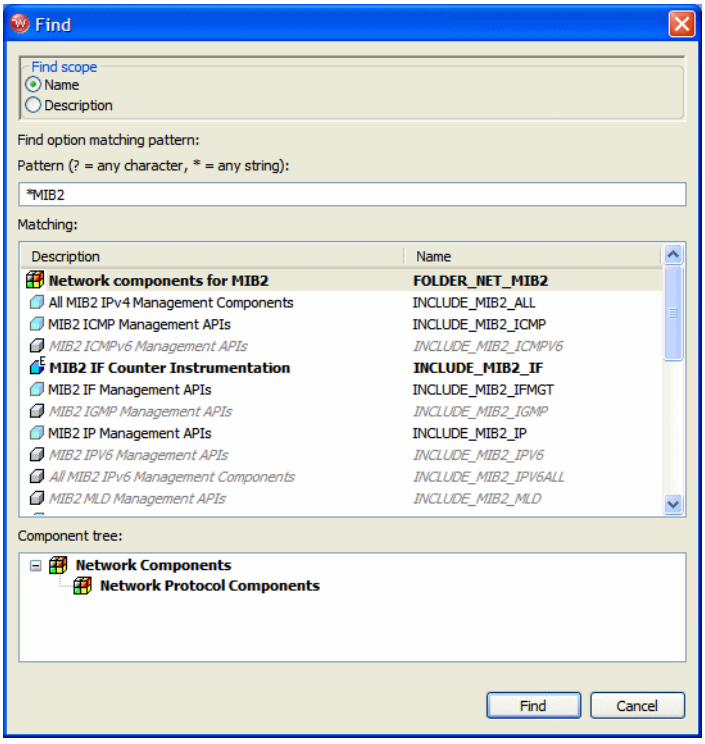
  – The component may not be installed.

4.2.2 Using Find to Locate Components

The Find tool searches component **INCLUDE**_*ABC* macro names or component descriptions. If you know the component name or description (or part of it), you can use the Find tool to locate and select that component. The example in Figure 4-3, shows the matches found using the pattern ***MIB2**.

With the Components tab of the Kernel Configuration Editor active (see Figure 4-2), select **Edit > Find/Replace** (or press **Ctrl+F**) to open the Find tool.

Figure 4-3 **Using Find**



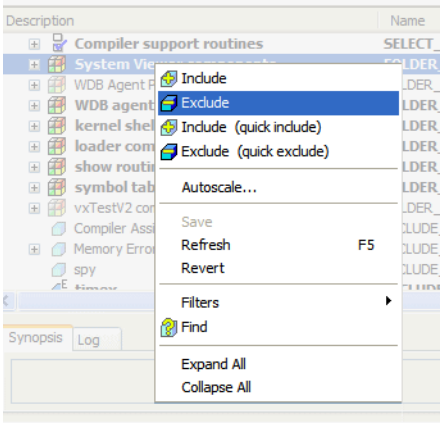
When you see the component you want, select it and click **Find**. This returns you to the Kernel Configuration Editor with the component highlighted.

You can also browse through the component hierarchy in the Kernel Configuration Editor.

For more information on using the Find tool, see *Wind River Workbench User Interface Reference: Kernel Configuration Editor*.

4.2.3 Including or Excluding Components

You can include or exclude components in the Kernel Configuration Editor, by right-clicking the component you wish to include or exclude and selecting the desired action from the context menu.



Workbench handles component dependencies for you. That is, any components dependent on the component you have selected will be included (or excluded) automatically.

The **Include** and **Exclude** options in the context menu launch an additional dialog that shows you which components will be included (or excluded) and allows you to apply specific changes to any subcomponents. The **Include (quick include)** and **Exclude (quick exclude)** options automatically include (or exclude) the default selections associated with the component.

Once you have made the desired changes, you must rebuild your VxWorks Image project.

For more information on including and excluding components using the Kernel Configuration Editor, see the *Wind River Workbench User Interface Reference: Kernel Configuration Editor*.

4.3 Managing Components Using a Command-Line Build

You can also include or exclude components using a command-line build. There are two methods for performing VxWorks command-line builds:

- You can add or remove components or component bundles using the **vxprj** command-line build facility.

- You can edit the definitions in your BSP **config.h** file.

For more information on which command-line build methods are available (or preferred) for a given component, see the documentation for that component.

vxprj Build Method

Using **vxprj** for your command-line build helps ensure that any component dependencies are accounted for and provides a simple command-line interface. For more information on using the **vxprj** command-line build facility, see *VxWorks Command-Line Tools User's Guide*.

BSP config.h Build Method

You can control your VxWorks image configuration by including or excluding definitions in the target-specific configuration header file **config.h**. This file is located in the directory *installDir*/**vxworks-6.3/target/config/***bspName*. This file contains definitions that apply only to a specific target, and can also redefine default definitions in **configAll.h** that are inappropriate for a particular target. For example, if a target cannot access a device controller at the default I/O address defined in **configAll.h** because of addressing limitations, the address can be redefined in **config.h**.

The **config.h** header file includes definitions for the following parameters:

- default boot parameter string for boot ROMs
- interrupt vectors for system clock and parity errors
- device controller I/O addresses, interrupt vectors, and interrupt levels
- shared memory network parameters
- miscellaneous memory addresses and constants

WARNING: Do *not* edit **configAll.h**.

Some platform components require manual editing beyond defining or undefining the component **INCLUDE**_*ABC* macro and supplying parameter macro values. For more information, see the individual component documentation.

4.4 Configuring OS Components

For details about configuring all of the VxWorks core OS run-time components, see *VxWorks Kernel Programmer's Guide*.

4.5 Configuring Connectivity Components

General Purpose Platform *connectivity* components include:

- Wind River Network Stack
- Wind River PPP
- Wind River TIPC
- Wind River USB

4.5.1 Wind River Network Stack

The Wind River Network Stack is a full-featured dual mode (IPv4 and IPv4/IPv6) TCP/IP stack. The network stack provides a wide array of networking libraries and applications, some of which can be configured in ways that may require recompilation of your source code. For more information on the network stack components, as well as additional configuration options, see the *Wind River Network Stack for VxWorks 6 Programmer's Guide*.

Step 1: Include the Network Stack Components

1. Right-click **Network Components** and select **Include**.
2. Select the features you want to include, then click **Finish**.

4.5.2 Wind River PPP

Wind River PPP provides a configurable framework for creating and managing various types of PPP connections. This section describes the basic steps required to implement a basic serial PPP connection framework. For full instructions for each connection type as well as instructions for creating and testing PPP connections, see *Wind River PPP for VxWorks 6 Programmer's Guide*.

Step 1: Increase the Value of IP_MAX_UNITS

IP_MAX_UNITS indicates the maximum number of IP interfaces that the network stack will support. Each PPP connection that you create (serial or PPP over Ethernet) requires one IP interface. Locate the **IP_MAX_UNITS** parameter with the Find tool.

Step 2: Include the DNS Resolver Component in Your Image

1. Include **INCLUDE_DNS_RESOLVER.**
2. Within this component, set the following parameters:

   RESOLVER_DOMAIN
   For example: **"wrs.com"**

   RESOLVER_DOMAIN_SERVER
   For example: **"90.0.0.11"**

   If you do not know the correct values for these parameters, consult your system administrator.

NOTE: The quotation marks in the examples above are part of the value. You must include these quotation marks in whatever values you supply for the **RESOLVER_DOMAIN** and **RESOLVER_DOMAIN_SERVER** parameters.

Step 3: Include Either the PPP Client or the PPP Server in Your Image

Add one of the following components:

INCLUDE_PPP_CLIENT
PPP client (DUN)—the PPP client

INCLUDE_PPP_RAS_CONC
Remote Access Server—the PPP server

If you are connecting two VxWorks targets, create two projects, one for each target. The kernel image for the client target should include the **INCLUDE_PPP_CLIENT** component. The kernel image for the server target should include **INCLUDE_PPP_RAS_CONC**.

Workbench adds a number of other necessary components to your project at this stage.

The default PPP protocol framework is referenced by the variable **pppSysFramework** and its base configuration profile by **pppBasicLinkProfile**. This profile contains basic configuration information for a PPP stack but does not include any port-specific information.

Use the **pfwComponentListShow( )** routine to generate a listing of those modules that have been included in this framework. For example, from the target shell issue the following command:

```
-> pfwComponentListShow pppSysFramework
```

Step 4: Configure the Serial Connections (Null Modem Cable and External Modem)

This procedure references configuration parameters and components such as **INCLUDE_PPP_COM***n* and **PPP_COM***n***_PEER_TYPE**. The *n* refers to the number of the COM port, that is: COM1 or COM2.

When selecting a COM port, keep in mind that if you include the target shell in your kernel image, COM1 is already taken (for the target shell) and is not available for PPP. If you create PPP connections on COM1, you must exclude the target shell from your kernel image. To do this, locate the parameter **CONSOLE_TTY** using the Find tool and change it from the default of COM1 (**0**) to NONE (**-1**).

1. Include the **basic configuration for serial port** *n* component.

   INCLUDE_PPP_COM*n*
   basic configuration for serial port *n*

   When you include this component, the Kernel Configuration Editor warns you of configuration errors. To fix this, you must set the value of the **PPP_COM***n***_PEER_TYPE** parameter as described in the next step.

2. Configure the **basic configuration for serial port** *n* component.

   Under the **Properties** tab, set the following parameters:

   PPP_COM*n***_PEER_TYPE**
   PPP_COM*n***_BAUDRATE**
   PPP_COM*n***_CLNT_AUTOSTART** (optional, for PPP clients only)

   Set the parameter values as follows:

   PPP_COM*n***_PEER_TYPE**
   Specify the type of connection by selecting one of the following values:

NOTE: When setting **PPP_COM***n***_PEER_TYPE**, the quotes around the value are part of the value. You must include them.

   "WIN_CLIENT"
   A Wind River PPP-server-to-Windows-Dial-up-Networking (DUN) connection over a null modem.

"WIN_SERVER"
A Wind River PPP-client-to-Windows-Remote-Access-Server (RAS) connection over a null modem.

"REGULAR"
A server-to-non-Windows or client-to-non-Windows connection (for instance Wind River PPP to Wind River PPP, Wind River PPP to Solaris, or Wind River PPP to Linux) over a null modem.

"MODEM"
A Wind River PPP-client-to-server (for instance Windows RAS) connection over an external modem.

This feature is supported in the demonstration client, which calls routines defined in: *installDir*/**vxworks-6.3/target/src/ppp/unsupported/pppModem.c**.

The demonstration client lets you dial out with **pppCom***n***Connect( )**, and hang up with **pppCom***n***Disconnect( )**. For details about these routines as well as the modem open, read, write, close, and **ioctl( )** interface, see *Wind River PPP for VxWorks 6 Programmer's Guide*.

The demonstration server does not handle modem (dial-in) connections. This is not a limit of Wind River PPP, just of the server demonstration code.

PPP_COM*n***_BAUDRATE**
If necessary, change the default value (**"19200"**) to match that of the peer. (The quotes are part of the value; you must include them.)

PPP_COM*n***_CLNT_AUTOSTART**
For a client image only, this specifies whether the client automatically connects at boot time. By default, the demonstration client is configured to automatically connect to the server at boot time. To prevent this, set this parameter to **FALSE**.

Note that if you prevent automatic boot-time connection, you must start the connection manually by calling **pppCom***n***Connect( )** after booting. For example:

```
-> pppCom2Connect
```

3. Configure IPv4 and IPv6 parameters.

Include the following component:

INCLUDE_COM*n***_IPV4_PARAMS**
ipv4 parameters for serial port *n*

If you will be running IPv6 over the PPP connection, also include:

INCLUDE_COM*n*_IPV6_PARAMS
ipv6 parameters for serial port *n*

4.5.3 Wind River TIPC

Wind River TIPC provides a variety of build options and configuration parameters. For detailed information on building and configuring your VxWorks image to include TIPC support, see *Wind River TIPC for VxWorks 6 Programmer's Guide: Building VxWorks to Include Wind River TIPC*.

Step 1: Include the TIPC Components

1. Right-click **Network Components > Network Protocol Components > TIPC components**, and select **Include**.
2. Select the features you want to include and click **Finish**.

Step 2: Include Show Routines (Optional)

If desired, include **Network Components > Network Utility Components > Show Routine Components > TIPC show routines**.

4.5.4 Wind River USB

Wind River USB provides both host controller components and peripheral (device) components. This section includes basic instructions for including USB components in a VxWorks Image project. The USB components are VxWorks kernel components.

USB Peripheral Stack Components

1. For a basic USB setup, include the USB peripheral stack component:
   - **INCLUDE_USB_TARG** (**USB Peripheral Stack**, required)

2. Include a target controller driver:
   - one of the following (required):
     - **INCLUDE_NET2280** (**NetChip NET2280**)
     - **INCLUDE_PDIUSBD12** (**PDIUSBD12**)
     - **INCLUDE_PHILIPS1582** (**PhilipsIsp1582**)
3. Include an appropriate function driver (optional):
   - **INCLUDE_KBD_EMULATOR** (**Keyboard Emulator**)
   - **INCLUDE_PRN_EMULATOR** (**Printer Emulator**)
   - **INCLUDE_MS_EMULATOR** (**Mass Storage Emulator**)—this also requires the dosFs file system component
4. To include device initialization at system startup, choose one or more of the appropriate device initialization components (optional):
   - **INCLUDE_KBD_EMULATOR_INIT** (**Targ Keyboard Emulator Init**)
   - **INCLUDE_PRN_EMULATOR_INIT** (**Targ Printer Emulator Init**)
   - **INCLUDE_MS_EMULATOR_INIT** (**Targ Mass Storage Emulator Init**)
5. To include a code exerciser, include the USB testing tool (optional):
   - **INCLUDE_USBTOOL** (**usbTool**)

   The USB testing tool performs all necessary USB driver initialization and therefore cannot be used if any of the initialization components (**INCLUDE**_*XXX*_**INIT** macros) are included.

USB Host Stack Components

This section describes the basic steps to include the USB host controller software components in your VxWorks Image project. Your choices will depend on the particular USB host controller(s) you plan to use. For additional details and descriptions of configuration parameters for each controller type, see *Wind River USB for VxWorks 6 Programmer's Guide*.

1. For a basic USB setup, include the USB host stack component (required):
   - **INCLUDE_USB** (**USB Host Stack**)

   Selecting the USB host stack component includes support for the USBD.

2. Include at least one of the host controller components:

   - **one or more of the following** (required):

     – **INCLUDE_EHCI** (**EHCI**)
     – **INCLUDE_OHCI** (**OHCI**)
     – **INCLUDE_UHCI** (**UHCI**)

   Selecting the **EHCI**, **OHCI**, or **UHCI** components includes modules for those types of host controllers, respectively. All host controller components require that the **USB Host Stack** component also be selected. More than one host controller can be present on the image at once.

3. To include initialization at system start for the USB host stack, include one or more of the host stack initialization component (optional):

   - **INCLUDE_USB_INIT** (**USB Host Stack Init**)

4. To include host controller initialization at system startup, include the appropriate host controller initialization components (optional):

   – **INCLUDE_EHCI_INIT** (**EHCI Init**)
   – **INCLUDE_OHCI_INIT** (**OHCI Init**)
   – **INCLUDE_UHCI_INIT** (**UHCI Init**)

5. If you intend to test a device, include one or more of the peripheral device components (optional):

   - **INCLUDE_USB_KEYBOARD** (**Keyboard**)
   - **INCLUDE_USB_MOUSE** (**Mouse**)
   - **INCLUDE_USB_PRINTER** (**Printer**)
   - **INCLUDE_USB_SPEAKER** (**Speaker**)
   - **INCLUDE_USB_MS_BULKONLY** (**Mass Storage - Bulk**)
   - **INCLUDE_USB_MS_CBI** (**Mass Storage - CBI**)
   - **INCLUDE_USB_PEGASUS_END** (**END - Pegasus**)

   Selecting any of the USB peripheral device components includes the corresponding driver module. These components require that the USB host stack be present in the VxWorks image.

6. To include device initialization at system startup, choose one or more of the appropriate device initialization component:
   - **INCLUDE_USB_KEYBOARD_INIT** (**Keyboard Init**)
   - **INCLUDE_USB_MOUSE_INIT** (**Mouse Init**)
   - **INCLUDE_USB_PRINTER_INIT** (**Printer Init**)
   - **INCLUDE_USB_SPEAKER_INIT** (**Speaker Init**)
   - **INCLUDE_MS_BULKONLY_INIT** (**Bulk Mass Storage Init**)
   - **INCLUDE_MS_CBI_INIT** (**CBI Mass Storage Init**)
   - **INCLUDE_USB_PEGASUS_END_INIT** (**END - Pegasus IPv4 Initialization**)

   Selecting any of the device initialization components includes the corresponding driver module. These components require that the USB host stack be present on the VxWorks image.
7. To include a code exerciser, include the USB testing tool (optional):
   - **INCLUDE_USBTOOL** (**usbTool**)

   The USB testing tool performs all necessary USB driver initialization. Therefore, it cannot be used if any of the initialization components (**INCLUDE**_*XXX*_**INIT** macros) are included. For more information about initialization and initialization dependencies, see *Wind River USB for VxWorks 6 Programmer's Guide*.

USB Components: Command-Line Configuration

USB components can be configured into a VxWorks image using a command-line build. For complete command-line configuration instructions, see *Wind River USB for VxWorks 6 Programmer's Guide.*

5 Migration

5.1 Introduction 47

5.2 Operating System Migration 48

5.3 Host Tool Migration 48

5.4 Connectivity Component Migration 49

5.1 Introduction

This chapter provides information for migrating to the Wind River General Purpose Platform, VxWorks Edition 3.3.

5.2 Operating System Migration

Migration and backwards compatibility issues for the transition from VxWorks 6.2 to VxWorks 6.3 are covered in detail in *VxWorks Migration Guide, 6.3*. In particular, critical information regarding file system and POSIX changes is available in the migration guide.

NOTE: Wind River recommends that all users migrating from earlier versions of VxWorks consult the migration guide before beginning development.

5.3 Host Tool Migration

This section documents host tool migration issues. Additional information regarding host tool migration issues is available in *Wind River Workbench Migration Guide*.

5.3.1 Wind River Workbench

Detailed information on migrating your existing applications and projects to Wind River Workbench 2.5 is available in the *Wind River Workbench Migration Guide.*

5.3.2 Wind River Compiler

The Wind River Compiler 5.4 is fully compatible with the previous release, Wind River Compiler 5.3.1.

5.3.3 Wind River GNU Compiler

The Wind River GNU Compiler 3.4.4 is fully compatible with the previous release, Wind River GNU Compiler 3.3.

5.3.4 Wind River ScopeTools

Wind River ScopeTools 5.5 are fully compatible with the previous release, Wind River ScopeTools 5.4.

5.4 Connectivity Component Migration

This section documents migration information for connectivity components. For more information on changes in these components, see the Platform release notes or the specific component documentation.

5.4.1 Wind River Network Stack

Wind River Network Stack 3.1 is fully compatible with the previous release, Wind River Network Stack 3.0.

5.4.2 Wind River PPP

Wind River PPP 2.2.2 is fully compatible with the previous release, Wind River PPP 2.2.1.

5.4.3 Wind River TIPC

Wind River TIPC 1.5 updates the previous release, Wind River TIPC 1.2, to provide full feature compatibility with open-source Linux TIPC 1.5. As a result, Wind River TIPC 1.5 makes major changes to the APIs and functionality provided with Wind River TIPC 1.2. These changes particularly affect TIPC message handling and subscriptions and add three new features: multicasting, stream socket support, and a TIPC configuration utility (**tipcConfig**).

In addition to new and modified features, Wind River TIPC 1.5 drops support for several show routines, API options, and configuration parameters.

To migrate a TIPC application from Wind River TIPC 1.2 to Wind River TIPC 1.5, you will probably need to rewrite a portion of your code. There are also changes in

build components and configuration parameters that you will need to be aware of. For detailed information on build, API, and configuration changes, see the Wind River TIPC chapter of the Platform release notes.

5.4.4 Wind River USB

The USB host stack now provides support for vendor-specific devices. Adding this feature required changing the interface definition of the **usbdDynamicAttachRegister( )** routine exposed by the translation unit. This routine now takes an additional parameter, **vendorSpecific**. The new interface is:

```
STATUS usbdDynamicAttachRegister
    (
    USBD_CLIENT_HANDLE clientHandle,     /* Client handle */
    UINT16 deviceClass,                  /* USB class/vendor code */
    UINT16 deviceSubClass,               /* USB sub-class/device code */
    UINT16 deviceProtocol,               /* USB device protocol/BCD code */
    BOOL vendorSpecific,                 /* For vendor specific devices */
                                         /* TRUE if vendor specific driver */
                                         /* FALSE if class specific driver */
    USBD_ATTACH_CALLBACK attachCallback  /* User-supplied callback  */
    )
```

For more information, see the release notes for Wind River USB and the **usbdDynamicAttachRegister( )** reference entry.

A WDB over TIPC

A.1 Introduction 51
A.2 Setup Requirements 52
A.3 Target Configuration 53
A.4 Establishing a Target Server Connection 54

A.1 Introduction

TIPC communication does not require any of the services provided by the UDP and TCP/IP protocols. Because of this, many nodes on a TIPC network (for example, VxWorks targets compiled with the **TIPC network stack** build option) lack the facilities to communicate directly with a development host outside of the TIPC network. In order to facilitate debugging and other host tool communication between a minimal TIPC node and Wind River Workbench, Wind River provides the WDB over TIPC feature for both VxWorks and Linux target operating systems.

This appendix provides a high level overview of the setup, configuration, and target connection process for WDB over TIPC for both VxWorks- and Linux-based systems. In particular, because a configuration where the TIPC network includes both VxWorks and Linux targets (for example, where a Linux target is the gateway and a VxWorks target compiled with the **TIPC network stack** build option is a TIPC node) is supported, this appendix illustrates how this process is parallel for

VxWorks and Linux targets. This appendix also includes references to additional documentation where you can go to find further information for both target types.

NOTE: This appendix does not provided detailed information for proper configuration and setup of WDB over TIPC on either VxWorks or Linux targets. You *must* review the documentation referenced in this appendix before configuring and running WDB over TIPC.

The WDB Target Agent

The WDB target agent is a run-time facility that allows you to connect Workbench or other host tools to a target using a target server. For more information on WDB, see *VxWorks Kernel Programmer's Guide: Target Tools* (VxWorks targets) or *Wind River Workbench User's Guide, Linux Edition: TIPC Configuration* (Linux targets).

TIPC

TIPC is a transparent inter-process communication (TIPC) infrastructure used for inter-node (cluster) communication. For more information on TIPC, see the open source TIPC Web site:

http://tipc.sourceforge.net/

For information on configuring and using Wind River TIPC with VxWorks, see *Wind River TIPC for VxWorks 6 Programmer's Guide*. For information on using TIPC with Linux targets, see *Wind River Workbench User's Guide, Linux Edition: TIPC Configuration*.

A.2 Setup Requirements

In order to use WDB over a TIPC network, you must set up your TIPC network according to some basic guidelines.

Setting Up Your TIPC Network

The following minimum setup for your TIPC network is required:

- You must have a minimum of two targets on your TIPC network and both targets must be located in the same zone and cluster.

- One of the targets on your TIPC network must act as a gateway. This target must have access to both the TIPC network and an IP network that is accessible to your host system.

This setup allows you to connect to the remaining target (or targets) on the TIPC network using WDB over TIPC.

A.3 Target Configuration

This section provides an overview of how to configure your gateway and TIPC nodes to use WDB over TIPC. For VxWorks targets, specific information regarding how to configure your VxWorks image is available in the *VxWorks Kernel Programmer's Guide: Target Tools* and the *Wind River TIPC for VxWorks 6 Programmer's Guide*. For Linux targets, see *Wind River Workbench User's Guide, Linux Edition: TIPC Configuration*.

Configuring the Gateway

The target you plan to use as a gateway to your TIPC network must include TIPC support as well as support for the WDB proxy agent. For VxWorks targets, you must include the **INCLUDE_WDB_PROXY** and **INCLUDE_WDB_PROXY_TIPC** components. For Linux targets, be sure you have installed the TIPC kernel module and use the **wrproxy** command to set up the WDB proxy agent.

Configuring the TIPC Nodes

To configure the target you want to connect to using WDB over TIPC, you must include and configure TIPC support on the target. You must also include a WDB agent and a WDB agent TIPC communication link for the target. For VxWorks targets, you can include the WDB agent TIPC communication link by including the **INCLUDE_WDB_COMM_TIPC** component in your VxWorks image or by setting **WDB_COMM_TYPE** to **WDB_COMM_TIPC** in your BSP **config.h** file. On Linux targets, launch the **usermode-agent** on the target you want to connect to.

NOTE: For VxWorks targets, TIPC nodes requiring WDB over TIPC support for host communication are configured with the **TIPC network stack** build option. For more information, see *Wind River TIPC for VxWorks 6 Programmer's Guide: Building VxWorks to Include Wind River TIPC*.

In addition to the above configuration, you must configure the WDB TIPC port type and the WDB TIPC port instance that you plan to use for WDB communication. You can accept the default values for these parameters or you can change the settings as necessary. For VxWorks targets, change the settings by altering the **WDB_TIPC_PORT_TYPE** and the **WDB_TIPC_PORT_INSTANCE** parameters (for more information, see *VxWorks Kernel Programmer's Guide: Target Tools*). For Linux targets, you can specify the WDB TIPC port type and port instance on the command line when starting the debug agent (for more information, see *Wind River Workbench User's Guide, Linux Edition*).

A.4 Establishing a Target Server Connection

To establish a target server connection to a target on your TIPC network, do the following:

1. Boot the targets on your network. This includes the target that will act as the TIPC gateway as well as the target you wish to connect to.
2. Start the target server on your host using the following command:

   -> **`tgtsvr -V -B wdbproxy -tipc -tgt`** *targetTipcAddr* **`-tipcpt`** *tipcPortType* **`-tipcpi`** *tipcPortInstance wdbProxyIPAddr*

   where *targetTipcAddr* is the TIPC address of the target you wish to connect to, *tipcPortType* is the TIPC port type used for the connection, *tipcPortInstance* is the TIPC port instance used for the WDB connection, and *wdbProxyIPAddr* is the IP address (or target name) of the target that is running the WDB proxy agent.

For more information on establishing a target server connection to a target on your TIPC network, see *Wind River Workbench User's Guide, VxWorks Edition: New Target Server Connections* (VxWorks targets) or *Wind River Workbench User's Guide, Linux Edition: TIPC Configuration* (Linux targets).

B

Removing Run-time Libraries

B.1 Introduction 55

B.2 Removing Run-time Libraries from VxWorks Images 55

B.1 Introduction

By default, both the Wind River Compiler and GNU GCC run-time libraries are included in the VxWorks image. If you want to remove either the Wind River Compiler or GNU GCC libraries, follow these instructions.

B.2 Removing Run-time Libraries from VxWorks Images

By default, VxWorks includes both **libc** and GNU **libgcc**, which are provided with the **INCLUDE_ALL_INTRINSICS** component. To exclude one library or the other, reconfigure the kernel with either **INCLUDE_DIAB_INTRINSICS** or **INCLUDE_GNU_INTRINSICS**, respectively.

NOTE: The **libc** and **libgcc** libraries are available in the kernel to enable dynamic downloading and running of kernel object modules.

Index

A

accessing the VxWorks development shell 13
applications
    RTP 16

B

boot ROM image
    building 13
build options
    TIPC network stack 51
building
    real-time processes (RTPs) 16
    run-time libraries 18
    shared libraries 17
    source build 18
    VxWorks OS source from the command line 25
    VxWorks OS source with Workbench 22

C

cmpscriptLib 15
command line
    building
        building a VxWorks kernel image with ROMFS 15
        real-time processes (RTPs) 17
        shared libraries 17
        VxWorks boot loaders 13
        VxWorks boot loaders using make 13
        VxWorks kernel applications 16
        VxWorks kernel images 15
    managing components 36
    setting the environment from 12, 25
    USB configuration 45
    VxWorks development shell 13
compatibility
    RTP application binaries 16
compiling
    VxWorks OS source 19
components
    configuring VxWorks 38
    including or excluding using the Kernel Configuration Editor 35
    locating using the Find tool 34
    managing 31
        using the command line 36
        using the config.h build method 37
        using vxprj 37
        with Workbench 32
    migration 47
    VxWorks 4

VxWorks simulator 4
Wind River Network Stack 4
Wind River PPP 4
Wind River TIPC 5
Wind River USB 5
config.h 37
configAll.h 37
configuration header files 37
configuring
a VxWorks image 31
PPP 38
the network stack 38
TIPC 42
USB 42
CONSOLE_TTY 40
CPU
ARMARCH5 28
ARMARCH6 28
MIPS32 29
MIPS64 29
PENTIUM 29
PENTIUM2 29
PENTIUM3 29
PENTIUM4 29
PPC32 29
SH32 29
SH7750 29
SIMLINUX 29
SIMNT 29
SIMPENTIUM 29
SIMSPARCSOLARIS 29
supported values 28
XSCALE 29

D

DNS resolver component 39
documentation
accessing online documentation 9
context sensitive help 9
Eclipse documentation 9
guide 5
ordering hardcopy 10
PDF 10
release information 5
terminology and conventions 1
tutorials 12

E

EHCI
host controller initialization 44
environment variables 12

F

files
configuration header 37
Find tool 34
framework
PPP demonstration framework 39

G

General Purpose Platform
components 4
GNU compiler
description 4

H

header files 37
host 1
host tools
GNU compiler 4
ScopeTools 4
System Viewer 4
Wind River Compiler 4
Workbench 4

I

INCLUDE_DNS_RESOLVER 39
INCLUDE_EHCI 44
INCLUDE_EHCI_INIT 44
INCLUDE_GCC_INTRINSICS 23
INCLUDE_KBD_EMULATOR 43
INCLUDE_KBD_EMULATOR_INIT 43
INCLUDE_MS_BULKONLY_INIT 45
INCLUDE_MS_CBI_INIT 45
INCLUDE_MS_EMULATOR 43
INCLUDE_MS_EMULATOR_INIT 43
INCLUDE_NET2280 43
INCLUDE_OHCI 44
INCLUDE_OHCI_INIT 44
INCLUDE_PDIUSBD12 43
INCLUDE_PHILIPS1582 43
INCLUDE_PPP_CLIENT 39
  connecting two VxWorks targets 39
INCLUDE_PPP_RAS_CONC 39
  connecting two VxWorks targets 39
INCLUDE_PRN_EMULATOR 43
INCLUDE_PRN_EMULATOR_INIT 43
INCLUDE_UHCI 44
INCLUDE_UHCI_INIT 44
INCLUDE_USB 43
INCLUDE_USB_INIT 44
INCLUDE_USB_KEYBOARD 44
INCLUDE_USB_KEYBOARD_INIT 45
INCLUDE_USB_MOUSE 44
INCLUDE_USB_MOUSE_INIT 45
INCLUDE_USB_MS_BULKONLY 44
INCLUDE_USB_MS_CBI 44
INCLUDE_USB_PEGASUS_END 44
INCLUDE_USB_PEGASUS_END_INIT 45
INCLUDE_USB_PRINTER 44
INCLUDE_USB_PRINTER_INIT 45
INCLUDE_USB_SPEAKER 44
INCLUDE_USB_SPEAKER_INIT 45
INCLUDE_USB_TARG 42
INCLUDE_USBTOOL 43, 45
INCLUDE_WDB_COMM_TIPC 53
INCLUDE_WDB_PROXY 53
INCLUDE_WDB_PROXY_TIPC 53
including
  USB device components 44
  USB host controller components 43, 44
  USB host stack components 43
  USB peripheral stack components 42
initializing the USB host stack 44
installation
  media 2
  optional CDs 3
  required CDs 2
installDir 2
ioctl( ) 41
IP_MAX_UNITS 39

K

Kernel Configuration Editor 32
  browsing 34
  including and excluding components 35

L

libprocfs.a 29
libtffs.a 29
libusb.a 29
libusb2.a 29

M

make 13
managing
  components 31
migration 47
minimal kernel configuration 14
MODEM
  PPP peer type 41

N

network stack
configuring 38
Wind River Network Stack description 4

O

OHCI
host controller initialization 44
online support site
tutorials 12
ordering hardcopy documents 10

P

pfwComponentListShow( ) 40
example 40
PPP
basic configuration for serial port n 40
configuring 38
creating connections on COM1 40
demonstration framework 39
DNS resolver component 39
INCLUDE_COMn_IPV4_PARAMS 41
INCLUDE_COMn_IPV6_PARAMS 42
INCLUDE_DNS_RESOLVER 39
INCLUDE_PPP_CLIENT 39
INCLUDE_PPP_COMn
configuration errors when including 40
INCLUDE_PPP_RAS_CONC 39
including the client or server 39
IP_MAX_UNITS 39
ipv4 parameters for serial port n 41
ipv6 parameters for serial port n 42
MODEM 41
PPP_COMn_BAUDRATE
changing to match the peer 41
PPP_COMn_CLNT_AUTOSTART
default is TRUE 41
PPP_COMn_PEER_TYPE 40
REGULAR 41
remote access server 39
RESOLVER_DOMAIN 39
RESOLVER_DOMAIN_SERVER 39
serial connections
configuring 40
WIN_CLIENT 40
WIN_SERVER 41
Wind River PPP description 4
PPP client (DUN) 39
pppBasicLinkProfile 39
pppCom2Connect( ) 41
pppComnConnect( )
modem dial-out 41
pppComnDisconnect( )
modem hang-up 41
pppModem.c 41
pppSysFramework 39
projects
boot loader 13
DKM 15
RTP 16
user-defined 18
VIP 14
VxWorks downloadable kernel module 15
VxWorks image 14
VxWorks real-time process 16
VxWorks ROMFS file system 14
VxWorks shared library 17

R

real-time processes (RTPs)
building 16
building with a shared library 16
compatibility with earlier releases 16
registry 2
REGULAR
PPP peer type 41
remote access server 39
RESOLVER_DOMAIN 39
RESOLVER_DOMAIN_SERVER 39
ROMFS
adding to a VxWorks kernel image 15

routines
ioctl( ) 41
pfwComponentListShow( ) 40
pppCom2Connect( ) 41
pppComnConnect( ) 41
pppComnDisconnect( ) 41
usbdDynamicAttachRegister( ) 50
RTPs
See real-time processes (RTPs)

S

ScopeTools
CoverageScope 4
description 4
MemScope 4
ProfileScope 4
StethoScope 4
TraceScope 4
setting up your TIPC network 52
shared libraries 17
building 17
building an RTP with 16
source code limitations
VxWorks OS source 20
System Viewer
description 4

T

target 1
target agent
WDB 52
target configuration
WDB over TIPC 53
target server 2
establishing a connection 54
TIPC 52
configuring 42
setting up your network 52
TIPC network stack build option 51
WDB over TIPC 51
Wind River TIPC description 5
tipcConfig 49
TOOL
supported values 28
tutorials 12

U

UHCI
host controller initialization 44
USB
command-line configuration 45
configuring 42
host controller
including host controller components 44
host controller initialization 44
host stack 43
host stack initialization 44
INCLUDE_EHCI 44
INCLUDE_EHCI_INIT 44
INCLUDE_KBD_EMULATOR 43
INCLUDE_KBD_EMULATOR_INIT 43
INCLUDE_MS_BULKONLY_INIT 45
INCLUDE_MS_CBI_INIT 45
INCLUDE_MS_EMULATOR 43
INCLUDE_MS_EMULATOR_INIT 43
INCLUDE_NET2280 43
INCLUDE_OHCI 44
INCLUDE_OHCI_INIT 44
INCLUDE_PDIUSBD12 43
INCLUDE_PHILIPS1582 43
INCLUDE_PRN_EMULATOR 43
INCLUDE_PRN_EMULATOR_INIT 43
INCLUDE_UHCI 44
INCLUDE_UHCI_INIT 44
INCLUDE_USB 43
INCLUDE_USB_INIT 44
INCLUDE_USB_KEYBOARD 44
INCLUDE_USB_KEYBOARD_INIT 45
INCLUDE_USB_MOUSE 44
INCLUDE_USB_MOUSE_INIT 45
INCLUDE_USB_MS_BULKONLY 44
INCLUDE_USB_MS_CBI 44
INCLUDE_USB_PEGASUS_END 44

INCLUDE_USB_PEGASUS_END_INIT 45
INCLUDE_USB_PRINTER 44
INCLUDE_USB_PRINTER_INIT 45
INCLUDE_USB_SPEAKER 44
INCLUDE_USB_SPEAKER_INIT 45
INCLUDE_USB_TARG 42
INCLUDE_USBTOOL 43, 45
including a code exerciser 43, 45
including a host controller 43
including a keyboard emulator 43
including a mass storage emulator 43
including a printer emulator 43
including device initialization at startup 43, 45
including the host stack 43
including the peripheral stack 42
including USB device components 44
peripheral stack 42
Wind River USB description 5
usbdDynamicAttachRegister( ) 50
use IPv6 enabled kernel libraries option 14
use source mode build option 14
use System Viewer free kernel libraries option 14
user-defined project 18
usermode-agent 53

V

variables
environment 12
vendorSpecific 50
vxprj 37
building a VxWorks kernel image 15
VxWorks
boot loader
building 13
building OS source from the command line 25
building OS source with Workbench 22
configuring 38
description 4
files supplied as object code only 20
kernel applications 15
kernel image 14
kernel image configuration 36
migration 47
OS source
building with a secondary compiler 23
compiling 19
restoring archives and object directories 28
supported CPU and TOOL values 28
ROMFS file system project
building 14
shared libraries 17
source code limitations 20
VxWorks boot loader projects 13
VxWorks downloadable kernel module projects 15
VxWorks image projects 14
kernel configuration 32
options 14
use IPv6 enabled kernel libraries 14
use source mode build 14
use System Viewer free kernel libraries 14
VxWorks kernel applications 15
VxWorks kernel images 14
building from the command line 15
configuring 31
configuring and building using the BSP config.h file 32
configuring and building with vxprj 32
configuring and building with Workbench 32
VxWorks real-time process projects 16
VxWorks shared library projects 17
VxWorks simulator
description 4

W

WDB
over TIPC 51
target agent 52
WDB_COMM_TIPC 53
WDB_COMM_TYPE 53
WDB_TIPC_PORT_INSTANCE 54
WDB_TIPC_PORT_TYPE 54
WIN_CLIENT
PPP peer type 40
WIN_SERVER
PPP peer type 41

Wind River Compiler
    description 4
Wind River registry 2
WIND_HOME 2
Workbench
    building
        real-time processes (RTPs) 16
        shared libraries 17
        VxWorks boot loaders 13
        VxWorks kernel applications 15
        VxWorks kernel images 14
        VxWorks ROMFS file system projects 14
    description 4
    Find tool 34
    managing components 32
    starting 12
wrenv
    overview 12
wrproxy 53